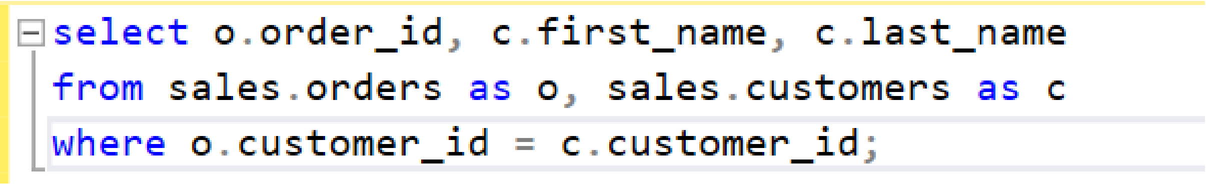
Technical Interview

1. SQL Interview Questions:

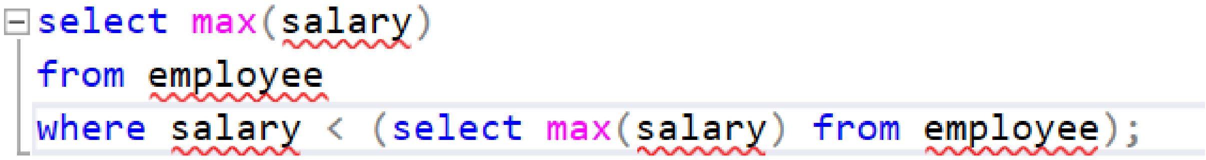
13. Given two tables, `orders` and `customers`, with relevant fields, write an SQL query to retrieve all orders along with the corresponding customer names.

Sol :



```
select o.order_id, c.first_name, c.last_name
from sales.orders as o, sales.customers as c
where o.customer_id = c.customer_id;
```

14. Using a subquery, find the second-highest salary from a table named `employees`.



```
select max(salary)
from employee
where salary < (select max(salary) from employee);
```

15. Explain what indexes are in the context of a database. How can you create an index on a specific column to improve query performance?

Sol: An index is a structure that holds the field the index is sorting and a pointer from each record to their corresponding record in the original table where the data is actually stored. Indexes are used in things like a contact list where the data may be physically stored in the order you add people's contact information but it is easier to find people when listed out in alphabetical order.

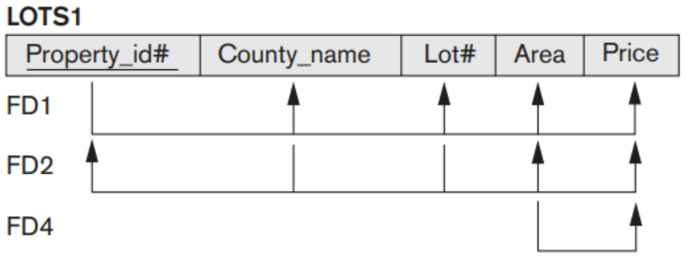
CREATE INDEX index_name ON table_name (column_name).

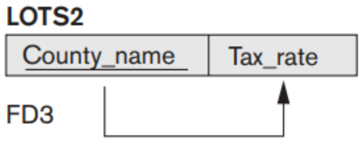
16. Describe the concept of database normalization. Provide an example of transforming an unnormalized table into third normal form (3NF).

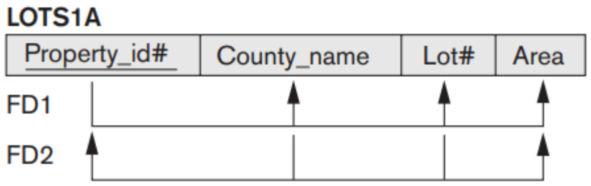
Sol: Normalization is a set of rules to systematically achieve a good design.

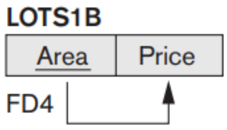
3NF is about removing the transitive dependencies by identifying the determinant of each transitive dependency, which then becomes a primary key of a new table and then assigning the appropriate attributes. An appropriate name of the new table needs to be chosen as well. BUT, the dependent attributes need to be removed from the original table, but the determinants remain and become foreign keys.

Example:



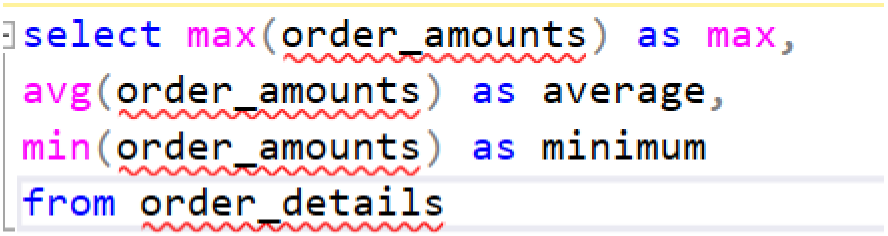






17. Write an SQL query to calculate the average, maximum, and minimum order amounts from an `order_details` table.

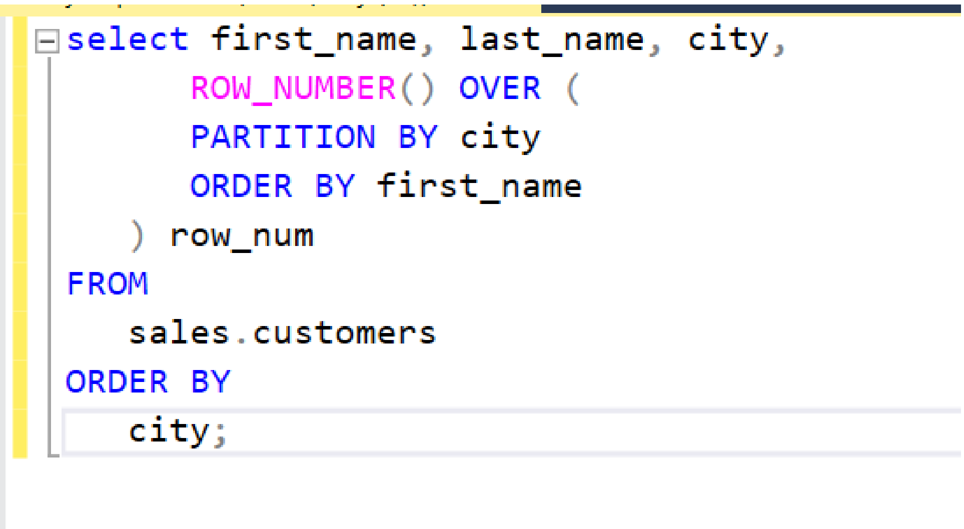
Sol :



```
select max(order_amounts) as max,
avg(order_amounts) as average,
min(order_amounts) as minimum
from order_details
```

18. Define window functions in SQL and provide an example of using the `ROW_NUMBER()` function to paginate results from a table.

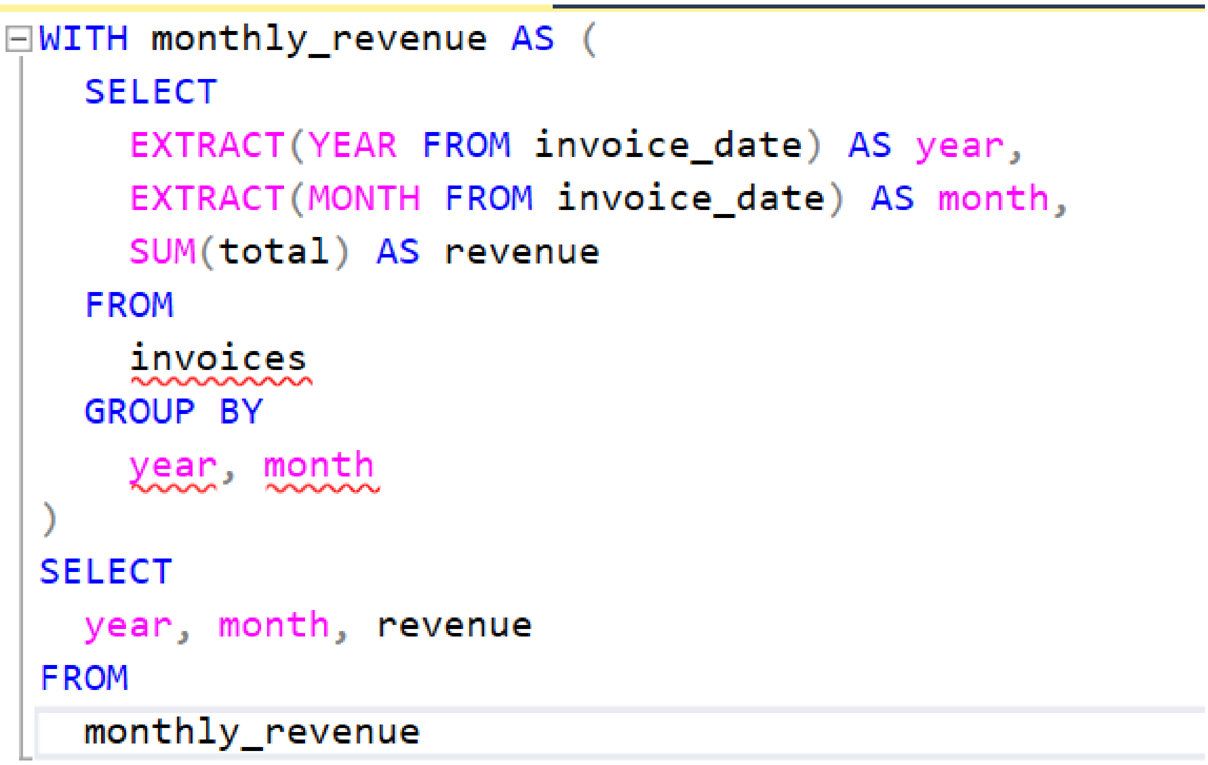
Sol: Window functions applies aggregate and ranking functions over a particular window (set of rows).The ROW_NUMBER() is a window function that assigns a sequential integer to each row within the partition of a result set. The row number starts with 1 for the first row in each partition.



```
select first_name, last_name, city,
        ROW_NUMBER() OVER (
        PARTITION BY city
        ORDER BY first_name
    ) row_num
FROM
    sales.customers
ORDER BY
    city;
```

19. Explain the purpose of Common Table Expressions (CTEs) in SQL. Create a CTE that calculates the total revenue for each month over a year from an `invoices` table.

Sol: CTEs are temporary named result sets that can be used within a query.CTEs enhance the flexibility, readability, and performance of SQL queries.



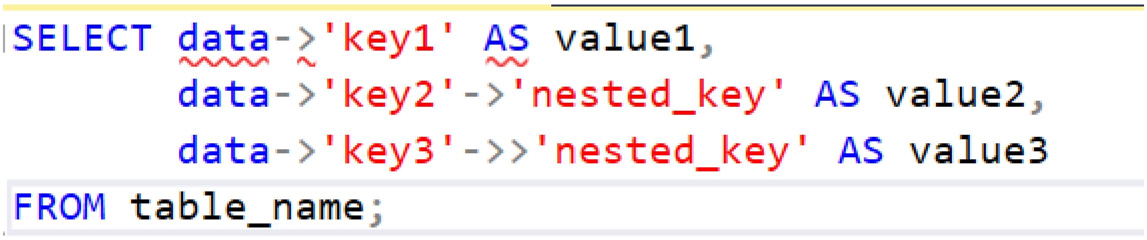
```
WITH monthly_revenue AS (
  SELECT
    EXTRACT(YEAR FROM invoice_date) AS year,
    EXTRACT(MONTH FROM invoice_date) AS month,
    SUM(total) AS revenue
  FROM
    invoices
  GROUP BY
    year, month
)
SELECT
  year, month, revenue
FROM
  monthly_revenue
```

20. Discuss what recursive queries are and when they might be used. Write a recursive SQL query to find all the ancestors of a given employee in an organizational hierarchy table.

Sol : Recursive queries are SQL queries that include self-referencing and iterative elements, allowing them to traverse hierarchical or recursive data structures. They used in Hierarchical Data, Graph Data, Recursive Calculations.

21. Describe the JSONB data type in PostgreSQL. Write a query that retrieves specific values from a JSONB column named `data` within a table.

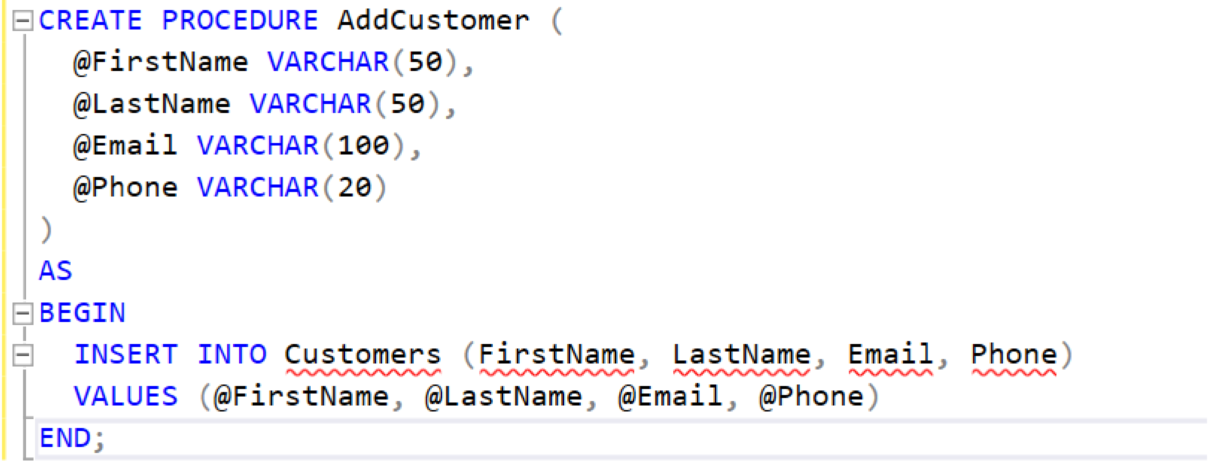
Sol: it is a data type in PostgreSQL that allows you to store and manipulate JSON data in a more effective and efficient way than the regular JSON data type.



```
SELECT data->'key1' AS value1,
       data->'key2'->'nested_key' AS value2,
       data->'key3'->>'nested_key' AS value3
FROM table_name;
```

22. How can stored procedures improve code organization and reusability in SQL? Provide an example of creating a stored procedure that inserts a new customer into a table.

Sol : Stored procedures in SQL can significantly improve code organization and reusability by encapsulating a set of SQL statements into a named procedure that can be invoked multiple times.



```
CREATE PROCEDURE AddCustomer (
    @FirstName VARCHAR(50),
    @LastName VARCHAR(50),
    @Email VARCHAR(100),
    @Phone VARCHAR(20)
)
AS
BEGIN
    INSERT INTO Customers (FirstName, LastName, Email, Phone)
    VALUES (@FirstName, @LastName, @Email, @Phone)
END;
```